

A Simple Model for Mutates

Google Ads API Migration Workshops - 2021



Thanet Knack Praneenararat, Developer Relations Engineer

Presenter

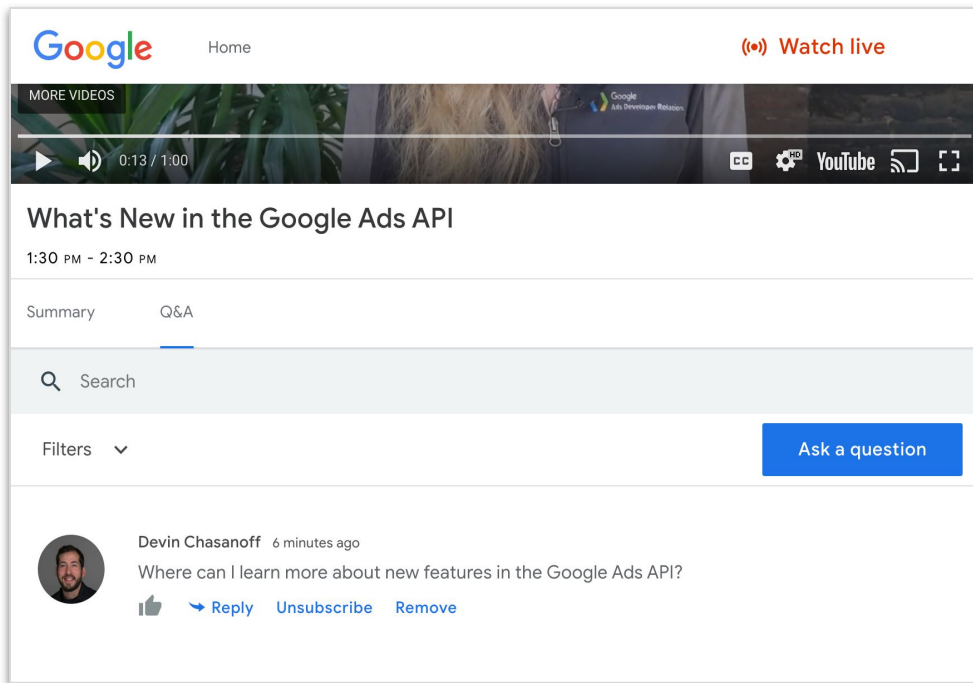


**Thanet Knack
Praneenarat**

Developer Relations Engineer

We're here to help!

- Q&A forum located below the video
- Our team is standing by to help answer your questions
- Submit questions at anytime
- Upvote interesting questions



The screenshot shows a YouTube video player interface. At the top, the Google logo is on the left, 'Home' is in the center, and a 'Watch live' button is on the right. Below the logo is a 'MORE VIDEOS' section with a video thumbnail. The video player shows a play button, a volume icon, and a progress bar at 0:13 / 1:00. To the right of the video player are icons for closed captions, settings, YouTube, and full screen. Below the video player is the video title 'What's New in the Google Ads API' and the time '1:30 PM - 2:30 PM'. There are two tabs: 'Summary' and 'Q&A', with 'Q&A' being the active tab. Below the tabs is a search bar with a magnifying glass icon and the text 'Search'. To the left of the search bar is a 'Filters' dropdown menu. To the right is a blue button labeled 'Ask a question'. Below the search bar is a Q&A post by 'Devin Chasanoff' posted '6 minutes ago'. The question is 'Where can I learn more about new features in the Google Ads API?'. Below the question are icons for a thumbs up, a reply arrow, and links for 'Reply', 'Unsubscribe', and 'Remove'.

Agenda

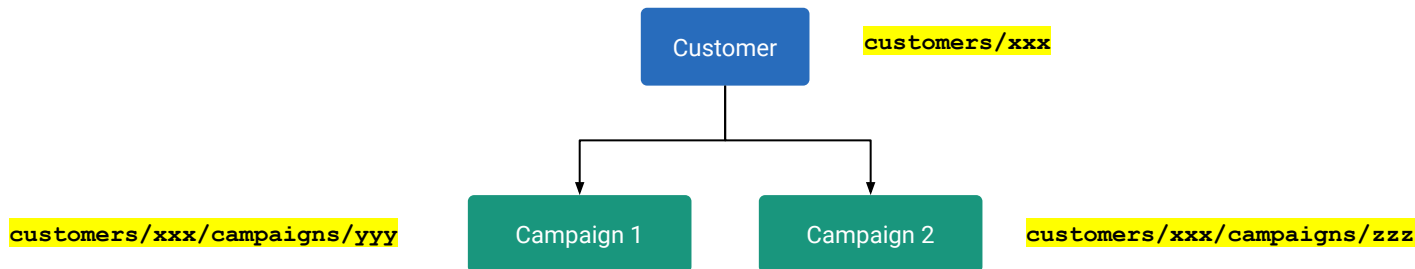
- Introduction to mutate
- Types of mutate methods
 - Single-resource mutate method
 - Multi-resource mutate method
 - Batch processing
- Features
 - FieldMask
 - Response content type
 - Temporary IDs
 - Mutate validation
- Sample use cases with code examples
- Caveats & special cases



Introduction to mutate

Modifying Google Ads API entities

- Google Ads API consists of **resources** and **services**
 - **Resource** represents a Google Ads entity and is identified by **resource names**



- **Services** modify (**create, update, remove**) Google Ads entities
 - For example, [CampaignService](#) for modifying campaigns
 - Method names are mostly in the form of Mutate<Resources>, e.g., [MutateCampaigns](#)
 - There are some exceptions, such as, [CreateCustomerClient](#)

Creating a new campaign using MutateCampaigns()

Proprietary + Confidential

PHP

```
$campaign = new Campaign([
    'name' => 'Interplanetary Cruise #' . uniqid(),
    'status' => CampaignStatus::PAUSED,
    'campaign_budget' => '<INSERT_CAMPAIGN_BUDGET_RESOURCE_NAME>',
    'advertising_channel_type' => AdvertisingChannelType::SEARCH,
    'manual_cpc' => new ManualCpc(),
]);
```

```
$campaignOperation = new CampaignOperation();
$campaignOperation->setCreate($campaign);
```

```
$campaignServiceClient = $googleAdsClient->getCampaignServiceClient();
$response = $campaignServiceClient->mutateCampaigns($customerId, [$campaignOperation]);
```

Output

Method

Input

Mutate input & output

- Methods for mutating entities have **one input** and **one output**
 - Input type: <MethodName>**Request**
 - Output type: <MethodName>**Response**
- Example of [MutateCampaigns](#):

Method	Input	Output
<code>rpc MutateCampaigns</code>	<code>(MutateCampaignsRequest)</code>	returns <code>(MutateCampaignsResponse)</code>

```
rpc MutateCampaigns(MutateCampaignsRequest) returns (MutateCampaignsResponse) {  
  option (google.api.http) = {  
    post: "/v8/customers/{customer_id=*/campaigns:mutate"  
    body: "*" }  
};  
option (google.api.method_signature) = "customer_id,operations";  
}
```

Proto3

Mutate input (request)

- Common properties of a mutate request are:

- `customer_id`

- `operations`

Most methods have these two properties

- `partial_failure`

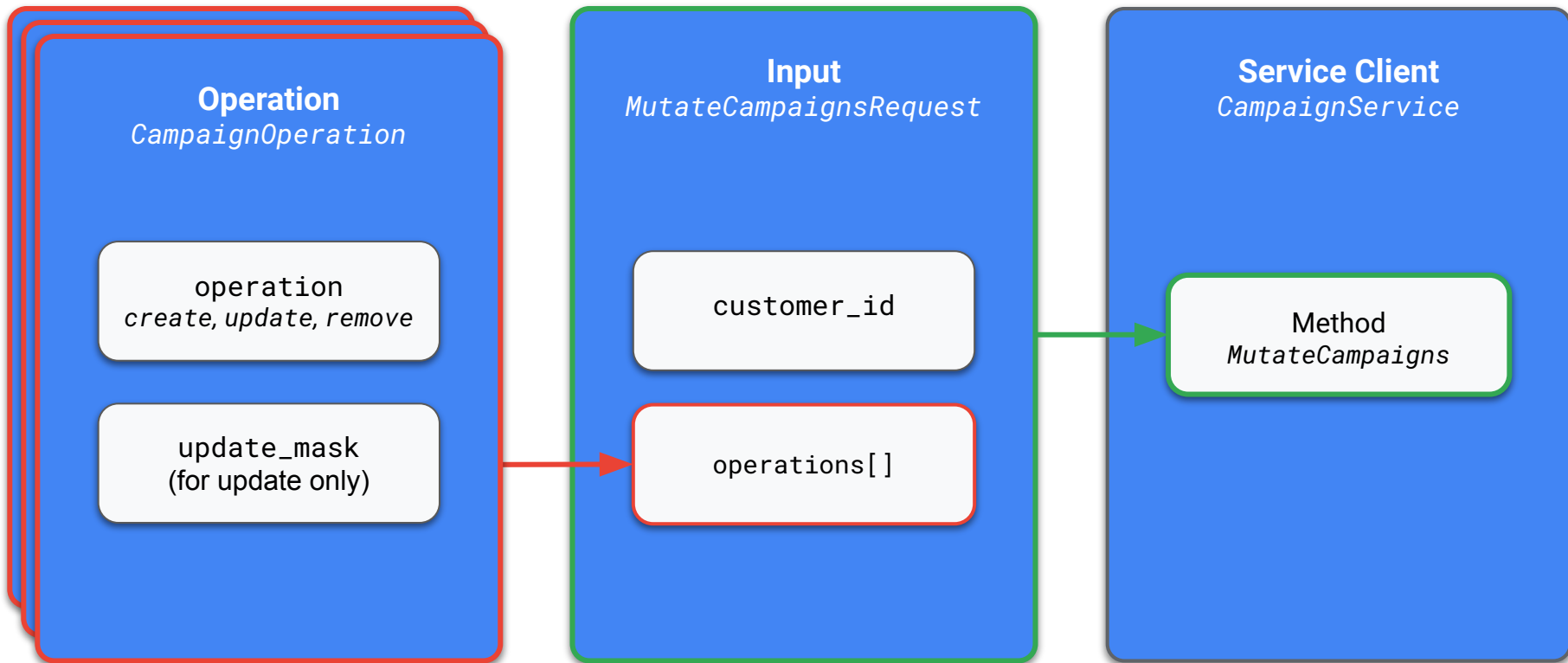
- `validate_only`

- `response_content_type`

More on this soon

- **Caveat:** *Not* all requests include the properties listed above

Mutate request (simplified diagram)



Note: partial_failure, validate_only and response_content_type are excluded from the **Input** for simplicity

MutateCampaignsRequest ([campaign_service.proto](#))

Proto3

```
message MutateCampaignsRequest {  
  string customer_id = 1 [(google.api.field_behavior) = REQUIRED];  
  repeated CampaignOperation operations = 2 [(google.api.field_behavior) = REQUIRED];  
  bool partial_failure = 3;  
  bool validate_only = 4;  
  google.ads.googleads.v8.enums.ResponseContentTypeEnum.ResponseContentType  
    response_content_type = 5;  
}
```

MutateCampaignsRequest ([campaign_service.proto](#))

Proto3

```
message MutateCampaignsRequest {  
  string customer_id = 1 [(google.api.field_behavior) = REQUIRED];  
  repeated CampaignOperation operations = 2 [(google.api.field_behavior) = REQUIRED];  
  bool partial_failure = 3;  
  bool validate_only = 4;  
  google.ads.googleads.v8.enums.ResponseContentTypeEnum.ResponseContentType  
    response_content_type = 5;  
}
```

MutateCampaignsRequest ([campaign_service.proto](#))

Proto3

```
message CampaignOperation {
```

```
  google.protobuf.FieldMask update_mask = 4; Used for selecting the fields to update
```

```
  oneof operation {
```

```
    google.ads.googleads.v8.resources.Campaign } create = 1; A campaign object is needed for "create" or "update"  
    google.ads.googleads.v8.resources.Campaign } update = 2;
```

```
    string remove = 3; Only resource name is needed for "remove"
```

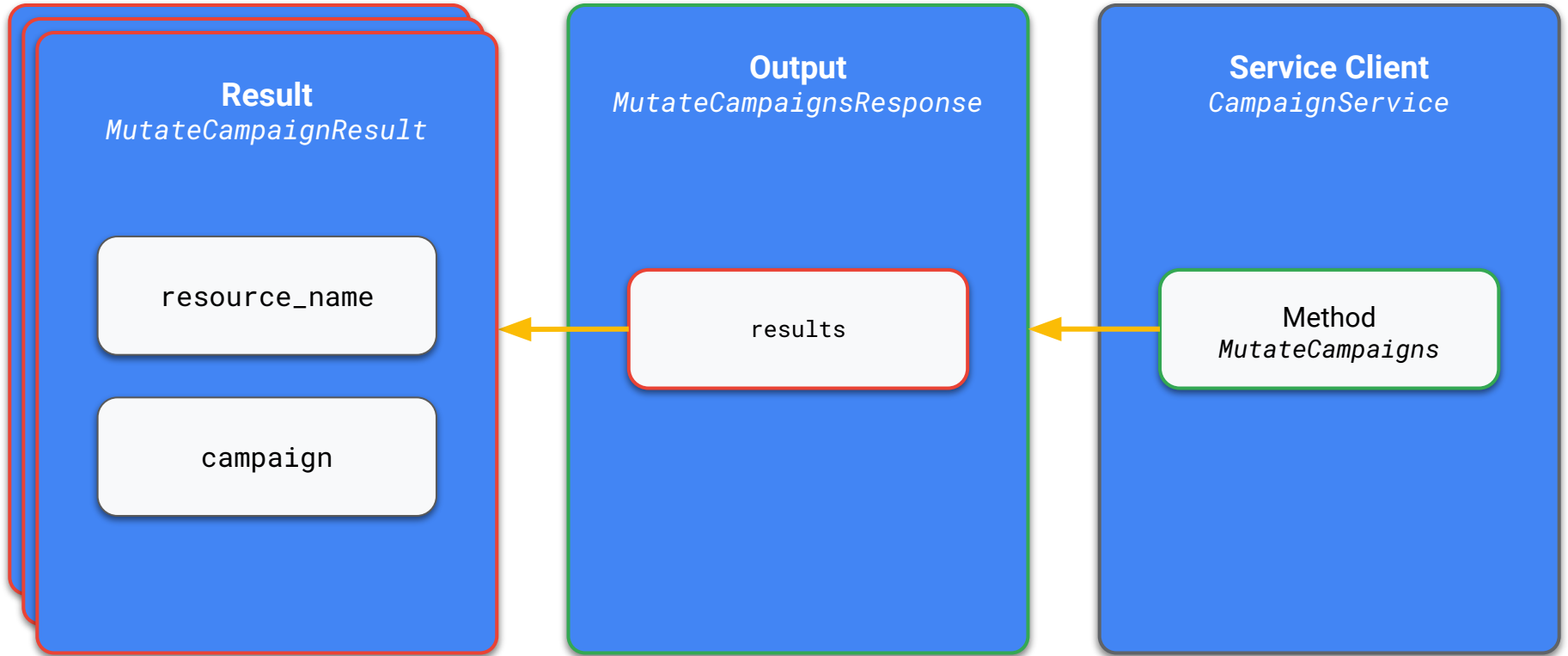
```
  }
```

```
}
```

Mutate output (response)

- Common properties of a mutate response are:
 - `partial_failure_error` **Errors when `partial_failure` is enabled in the request**
 - `results` **Most methods have this form of output**
- **Caveat:** *Not* all responses have exactly all of the above properties

Mutate response



Note: campaign is *not always* returned by default. More on this soon.

MutateCampaignsResponse ([campaign_service.proto](#))

Proto3

```
message MutateCampaignsResponse {  
  google.rpc.Status partial_failure_error = 3;  
  repeated MutateCampaignResult results = 2;  
}
```

```
message MutateCampaignResult {
```

```
  string resource_name = 1;
```

Most common output of mutate requests

```
  // The mutated campaign with only mutable fields after mutate. The field will  
  // only be returned when response_content_type is set to "MUTABLE_RESOURCE".  
  google.ads.googleads.v8.resources.Campaign campaign = 2;
```

```
}
```


MutateCampaignsResponse ([campaign_service.proto](#))

Proto3


```
message MutateCampaignsResponse {  
  google.rpc.Status partial_failure_error = 3;  
  repeated MutateCampaignResult results = 2;  
}
```

```
message MutateCampaignResult {  
  string resource_name = 1;  
  // The mutated campaign with only mutable fields after mutate. The field will  
  // only be returned when response_content_type is set to "MUTABLE_RESOURCE".  
  google.ads.googleads.v8.resources.Campaign campaign = 2;  
}
```

Most common output of mutate requests

Types of mutate methods

Types of mutate methods

- **Single-resource:** similar to *regular mutate methods* in AdWords API
- **Multi-resources:** *more powerful* than single-resource but *limited* 
- **Batch processing:** similar to *batch processing* in AdWords API



Single-resource mutate method

Single-resource mutate method

Method name: Such as, [CampaignService.MutateCampaigns](#)

Sample use cases

- Creating a few several campaigns that *share the same budget*
- Removing thousands of ad group criteria

Pros

- **Similar** to AdWords API
- **Every** resource has a corresponding mutate service

Cons

- Can easily lead to **orphaned resources**
- Error handling is **fully manual**
- **Maximum timeout** of 60 seconds (server)

Example: MutateCampaigns RPC ([campaign_service.proto](#))

Proto3

```
// Creates, updates, or removes campaigns. Operation statuses are returned.
rpc MutateCampaigns(MutateCampaignsRequest) returns (MutateCampaignsResponse) {
  option (google.api.http) = {
    post: "/v8/customers/{customer_id=*/campaigns:mutate"
    body: "*"
  };
  option (google.api.method_signature) = "customer_id,operations";
}
```

Multi-resource mutate method

Multi-resource mutate method



Method name: [GoogleAdsService.Mutate](#)

Sample use cases

- Creating a new campaign and multiple ad groups belonging to that campaign in one request

Pros

- **Single service** for *most* mutations
- Can include **different** operation and resource types
- **Avoid orphaned resources**
- Use of [temporary IDs](#) **→ More on this soon**

Cons

- **Not all** resource types are supported
- Error handling is **fully manual**
- **Maximum timeout** of 60 seconds (server)

Mutate RPC ([google_ads_service.proto](#))

**NEW****Proto3**

```
rpc Mutate(MutateGoogleAdsRequest) returns (MutateGoogleAdsResponse) {  
  option (google.api.http) = {  
    post: "/v8/customers/{customer_id=*/googleAds:mutate"  
    body: "*"   
  };  
  option (google.api.method_signature) = "customer_id,mutate_operations";  
}
```

MutateGoogleAdsRequest ([google_ads_service.proto](#))

NEW

Proto3

```
message MutateGoogleAdsRequest {
  string customer_id = 1
    [(google.api.field_behavior) = REQUIRED];
  repeated MutateOperation mutate_operations = 2
    [(google.api.field_behavior) = REQUIRED];
  bool partial_failure = 3;
  bool validate_only = 4;
  google.ads.googleads.v8.enums.ResponseContentTypeE
num.ResponseContentType
  response_content_type = 5;
}
```

```
message MutateOperation {
  // The mutate operation.
  → oneof operation {
    // An ad group ad mutate operation.
    AdGroupAdOperation ad_group_ad_operation = 1;
    // An ad group mutate operation.
    AdGroupOperation ad_group_operation = 5;
    // A campaign mutate operation.
    CampaignOperation campaign_operation = 10;
    // Other operations for supported resources.
    // ...
  }
}
```

MutateGoogleAdsResponse ([google_ads_service.proto](#))

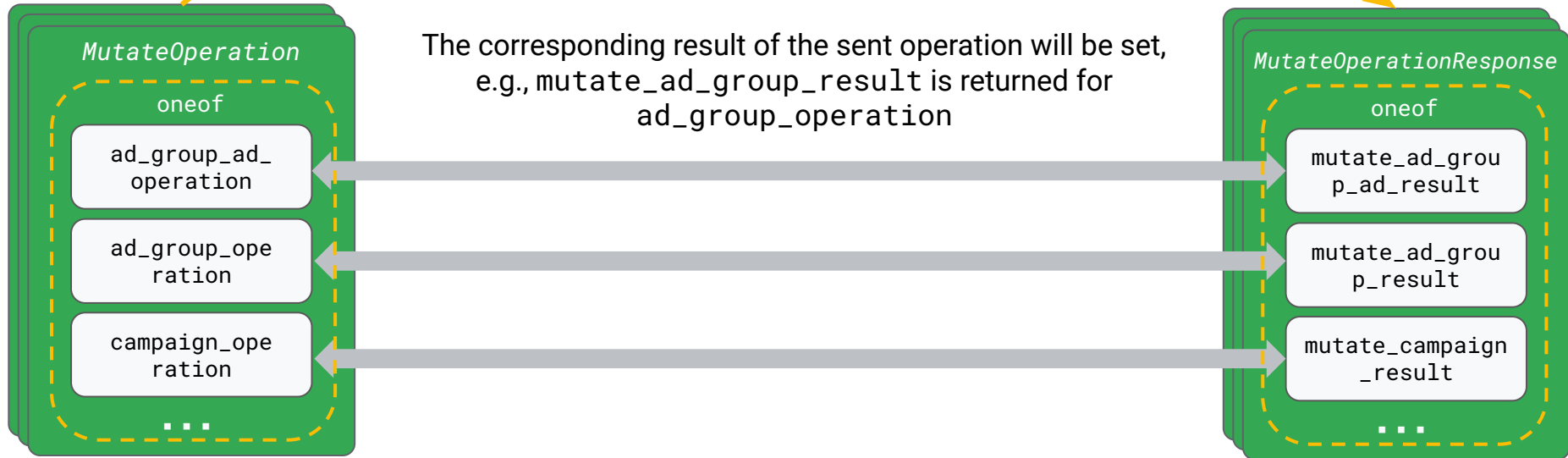
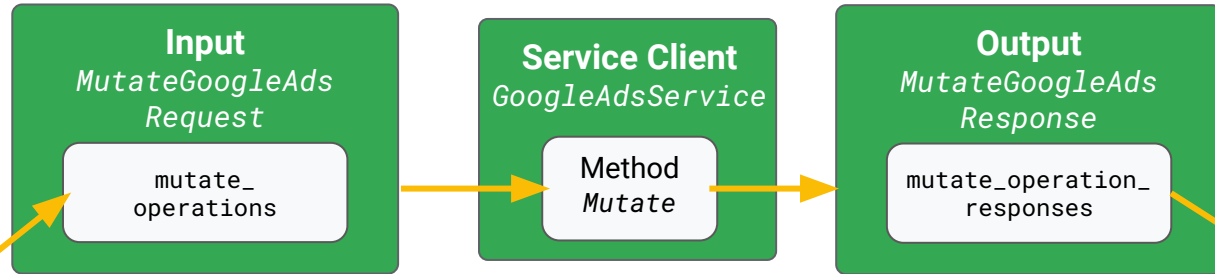
NEW

Proto3

```
message MutateGoogleAdsResponse {  
  google.rpc.Status partial_failure_error = 3;  
  repeated MutateOperationResponse  
    mutate_operation_responses = 1;  
}
```

```
message MutateOperationResponse {  
  // The mutate response.  
  → oneof response {  
    // The result for the ad group ad mutate.  
    MutateAdGroupAdResult ad_group_ad_result = 1;  
    // The result for the ad group mutate.  
    MutateAdGroupResult ad_group_result = 5;  
    // The result for the campaign mutate.  
    MutateCampaignResult campaign_result = 10;  
    // Other results for supported resources.  
    // ...  
  }  
}
```

Operation - result relationship



Batch processing


Batch processing

Method name: many methods of [BatchJobService](#)

Sample use cases

- Pausing 500,000 ad group ads
- Creating many unrelated sets of dependent objects (e.g., many Shopping listing groups)

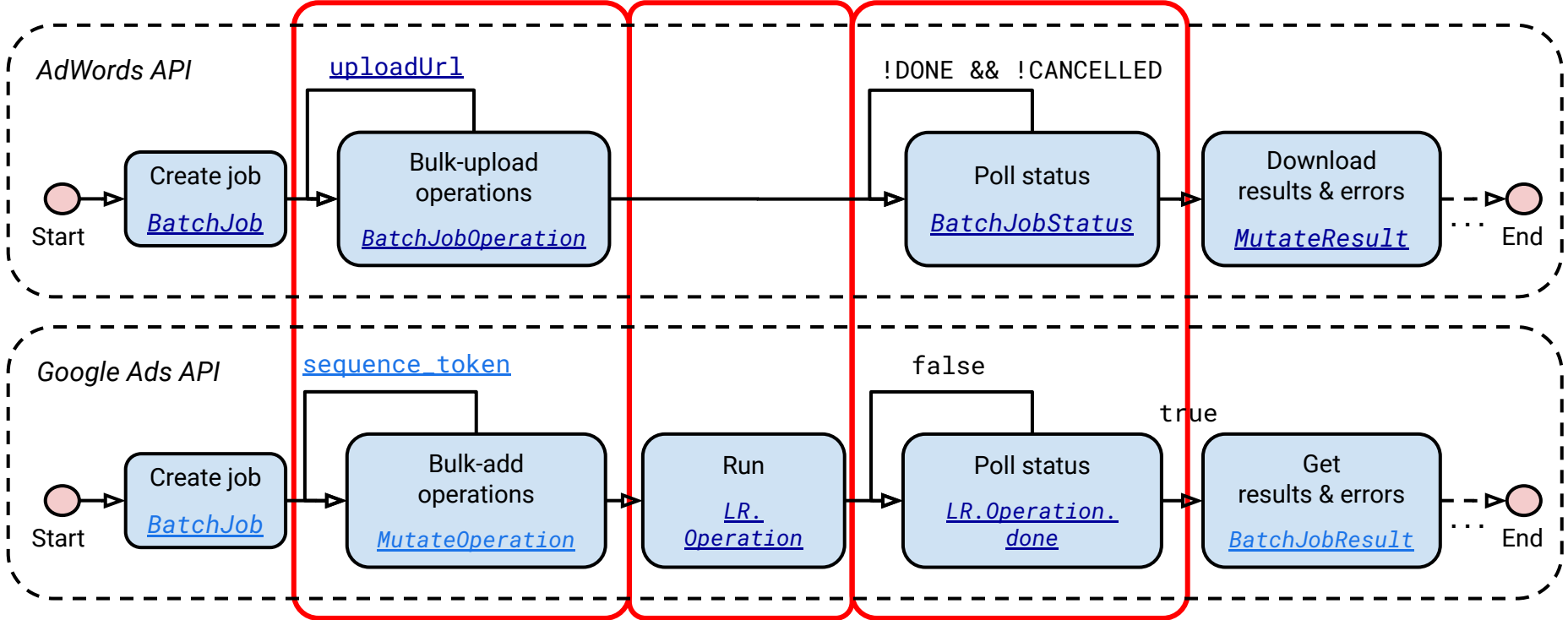
Pros

- **Resilient:** server error handling
- Can include **different** operation and resource types
- **Single service** for most mutations
- Use of [temporary IDs](#)  **More on this soon**
- Does not require good network connectivity


Cons

- **Not all resource types** are supported
- **Not sure when** it will finish (need to poll the result)
- The sequence flow is **more complex** with many steps

Usage



Recap - mutate method types

	Single-resource	Multi-resource 	Batch processing
Service	e.g., CampaignService.MutateCampaigns	GoogleAdsService.Mutate	BatchJobService
Request	5,000 operations*	5,000 operations*	1 million operations
Entity Mixture	One resource type	Mixed resource types	Mixed resource types
Timing	Synchronous	Synchronous	Asynchronous
Resource	All	Subset	Subset
Timeout	60 seconds	60 seconds	Unlimited
Scenario	Grouped operations	Grouped operations	Large processing, concurrent jobs
Error	Fully manual	Fully manual	Semi-automatic (transient)

* with some exceptions e.g., 1 for billing setup



Features



FieldMask

FieldMask

- Used for **identifying fields** you want to **update** for a given object
 - Those not in the field mask will be *ignored* even if they're set in the object
- Recommend using the **FieldMask utility** available in our [client libraries](#)
 - **compare(original, modified)** for creating a field mask based on *different field values*
 - **allSetFieldsOf(modified)** for creating a field mask based on *all fields that are set*

Campaign

```
{
  resource_name: "customers/xxx/campaigns/yyy",
  name: 'Test campaign',
  status: PAUSED,
  advertising_channel_type: HOTEL
  tracking_url_template: 'http://example.com'
}
```

FieldMask

- Used for **identifying fields** you want to **update** for a given object
 - Those not in the field mask will be *ignored* even if they're set in the object
- Recommend using the **FieldMask utility** available in our [client libraries](#)
 - **compare(original, modified)** for creating a field mask based on *different field values*
 - **allSetFieldsOf(modified)** for creating a field mask based on *all fields that are set*

Campaign

```
{
  resource_name: "customers/xxx/campaigns/yyy",
  name: 'Test campaign',
  status: PAUSED,
  advertising_channel_type: HOTEL
  tracking_url_template: 'http://example.com'
}
```

FieldMask

Operation for update

```
campaign_operation: {  
  update_mask: {  
    paths: ['name', 'status'] ← Only name and status will be updated  
  }  
  update: {  
    resource_name: "customers/xxx/campaigns/yyy", ← This field is always taken into account even  
    name: 'Test campaign',                               when it doesn't appear in the update_mask  
    status: PAUSED,  
    advertising_channel_type: HOTEL  
    tracking_url_template: 'http://example.com'  
  }  
}
```

These are ignored on the server side

Response content type

Response content type

[campaign_service.proto](#)

```
message MutateCampaignsRequest {
  string customer_id = 1 [(google.api.field_behavior) = REQUIRED];
  repeated CampaignOperation operations = 2 [(google.api.field_behavior) = REQUIRED];
  bool partial_failure = 3;
  bool validate_only = 4;
  google.ads.googleads.v8.enums.ResponseContentTypeEnum.ResponseContentType
    response_content_type = 5;
}
```

Proto3

- RESOURCE_NAME_ONLY
- MUTABLE_RESOURCE

- By default, the **resource names** of modified entities are returned
 - Equivalent to `response_content_type = RESOURCE_NAME_ONLY`
- Set `response_content_type` to [MUTABLE_RESOURCE](#) to get the values of all *mutable fields* for every object created / updated by the request
 - Use this to avoid an additional search or `searchStream` request
 - **Caution:** This takes **much more time** to finish. Use it only when the returned object will be used

Response content type

[campaign_service.proto](#)

Proto3

```
message MutateCampaignsResponse {  
  google.rpc.Status partial_failure_error = 3;  
  repeated MutateCampaignResult results = 2;  
}
```

```
message MutateCampaignResult {  
  string resource_name = 1; — Always returned regardless of response_content_type  
  // The mutated campaign with only mutable fields after mutate. The field will  
  // only be returned when response_content_type is set to "MUTABLE_RESOURCE".  
  google.ads.googleads.v8.resources.Campaign campaign = 2; — response_content_type = MUTABLE_RESOURCE  
}
```

response_content_type = MUTABLE_RESOURCE

Caveat: Only fields you modified (created / updated) + resource_name are populated

Temporary IDs

Temporary IDs: placeholder for future references

- A **negative ID** that you specify in the resource name of a new entity that can be used later to create other entities that *depend on this entity*

```
mutate_operations: [  
  {  
    campaign_operation: {  
      create: {  
        resource_name: "customers/<YOUR_CUSTOMER_ID>/campaigns/-1",  
        ...  
      }  
    }  
  },  
  {  
    ad_group_operation: {  
      create: {  
        campaign: "customers/<YOUR_CUSTOMER_ID>/campaigns/-1"  
        ...  
      }  
    }  
  }  
]
```

Temporary IDs will be assigned the real IDs by the Google Ads API server

New

Reference

Temporary IDs: rules

- Available when using **multi-resource mutate** and **batch processing**
- **The order of operations is important**
 - For example, the ad group operation would have to appear after the campaign operation
- Temporary IDs *cannot* be used across jobs or mutate requests
 - Use the real IDs instead

Temporary IDs: uniqueness of IDs

- Temporary IDs must be **unique** negative numbers, even if they are from *different* resource types
 - For example, temporary IDs for campaigns and ad groups must be different
 - Use a function or global variable that decrements each time it's used

Operation Type	Operation	Created Resource Name <small>*All resource names in this column must be unique</small>	Referenced Resource Name
campaign_operation	create	/customers/[INSERT_ID]/campaigns/-1	
ad_group_operation	create	/customers/[INSERT_ID]/adGroups/-2	/customers/[INSERT_ID]/campaigns/-1
ad_group_ad_operation	create		/customers/[INSERT_ID]/adGroups/-2

Mutate validation

Mutate validation

- Most mutate requests can be **validated** before executing the real call
- Set the request's **validate_only** to **true**
 - The request is validated, but the final execution is skipped
 - If no errors are found, an **empty response** is returned
 - If validation fails, **error messages** will be returned as usual
- Particularly useful in testing ads for **common policy violations**
 - **Caveat:** Errors can still occur in the real calls even if you get no errors in the validation mode

[ad_group_ad_service.proto](#)

```
message MutateAdGroupAdsRequest {  
  string customer_id = 1 [(google.api.field_behavior) = REQUIRED];  
  repeated AdGroupAdOperation operations = 2 [(google.api.field_behavior) = REQUIRED];  
  bool validate_only = 4;  
  // ...  
}
```

Proto3

Sample use cases with code examples

Sample use cases with code examples

- **Single-resource mutate:** Creating several campaigns sharing the same budget
- **Multi-resource mutate:** Creating a new campaign and multiple ad groups of the campaign in one request
- **Batch processing:** Pausing 500,000 ad group ads

Single-resource mutate's code example

Creating several campaigns sharing the same budget

Proprietary + Confidential

(Based on [AddCampaigns](#))

```
$campaignOperations = [];  
for ($i = 0; $i < 3; $i++) {  
    $campaign = new Campaign([  
        'name' => 'Interplanetary Cruise #' . uniqid(),  
        'status' => CampaignStatus::PAUSED,  
        'campaign_budget' => $budgetResourceName,  
        'advertising_channel_type' => AdvertisingChannelType::SEARCH,  
        'manual_cpc' => new ManualCpc(),  
    ]);  
    $campaignOperations[] = new CampaignOperation([  
        'create' => $campaign  
    ]);  
}  
$campaignServiceClient = $googleAdsClient->getCampaignServiceClient();  
$response = $campaignServiceClient->mutateCampaigns($customerId, $campaignOperations);
```

3) Loop to create three campaigns

1) Create a new campaign object

2) Create a new campaign operation

4) Send a mutate request to the server

PHP

Creating several campaigns sharing the same budget

Proprietary + Confidential

(Based on [AddCampaigns](#))

PHP

```
$campaign = new Campaign([
    'name' => 'Interplanetary Cruise #' . uniqid(),
    'status' => CampaignStatus::PAUSED,
    'campaign_budget' => $budgetResourceName,
    'Advertising_channel_type'
        => AdvertisingChannelType::SEARCH,
    'manual_cpc' => new ManualCpc(),
]);
```

- **Budget with the resource name `$budgetResourceName` is shared among newly created campaigns**
- **Only campaigns are created here, so no concerns about orphan budgets**

Creating several campaigns sharing the same budget

(Based on [AddCampaigns](#))

Proprietary + Confidential

PHP

```
$campaignOperations[] = new CampaignOperation([  
    'create' => $campaign  
]);
```

**One CampaignOperation for
each new campaign**

Creating several campaigns sharing the same budget

(Based on [AddCampaigns](#))

Proprietary + Confidential

PHP

```
for ($i = 0; $i < 3; $i++) {
    $campaign = new Campaign([
        'name' => 'Interplanetary Cruise #' . uniqid(),
        'status' => CampaignStatus::PAUSED,
        'campaign_budget' => $budgetResourceName,
        'advertising_channel_type' => AdvertisingChannelType::SEARCH,
        'manual_cpc' => new ManualCpc(),
    ]);
    $campaignOperations[] = new CampaignOperation([
        'create' => $campaign
    ]);
}
```

Creating several campaigns sharing the same budget

(Based on [AddCampaigns](#))

Proprietary + Confidential

PHP

- One single-resource mutate request is sent to one service
- One request can contain *several operations*

```
$campaignServiceClient = $googleAdsClient->getCampaignServiceClient();  
$response = $campaignServiceClient->mutateCampaigns($customerId, $campaignOperations);
```

Multi-resource mutate's code example

Creating a new campaign and multiple ad groups of the campaign in one request

PHP

```
$mutateOperations = [];  
$campaignBudgetOperation = self::buildCampaignBudgetOperation($customerId);  
$mutateOperations[] = new MutateOperation(['campaign_budget_operation' => $campaignBudgetOperation]);  
  
$campaignOperation = self::buildCampaignOperation(  
    $customerId,  
    $campaignBudgetOperation->getCreate()->getResourceName()  
);  
$mutateOperations[] = new MutateOperation(['campaign_operation' => $campaignOperation]);  
  
$adGroupOperations = self::buildAdGroupOperations($campaignOperation->getCreate()->getResourceName());  
$mutateOperations = array_merge($mutateOperations, array_map(  
    function (AdGroupOperation $adGroupOperation) {  
        return new MutateOperation(['ad_group_operation' => $adGroupOperation]);  
    },  
    $adGroupOperations  
));  
  
$googleAdsServiceClient = $googleAdsClient->getGoogleAdsServiceClient();  
$response = $googleAdsServiceClient->mutate($customerId, $mutateOperations);
```


Creating a new campaign and multiple ad groups of the campaign in one request

PHP

1) Create a MutateOperation for a CampaignBudgetOperation

```
$mutateOperations = [];  
$campaignBudgetOperation = self::buildCampaignBudgetOperation($customerId);  
$mutateOperations[] = new MutateOperation(['campaign_budget_operation' => $campaignBudgetOperation]);
```

```
$campaignOperation = self::buildCampaignOperation(  
    $customerId,  
    $campaignBudgetOperation->getCreate()->getResourceName()  
);  
$mutateOperations[] = new MutateOperation(['campaign_operation' => $campaignOperation]);
```

2) Create a MutateOperation for a CampaignOperation

```
$adGroupOperations = self::buildAdGroupOperations($campaignOperation);  
$mutateOperations = array_merge($mutateOperations, array_map(  
    function (AdGroupOperation $adGroupOperation) {  
        return new MutateOperation(['ad_group_operation' => $adGroupOperation]);  
    },  
    $adGroupOperations  
));
```

3) Create MutateOperation objects for AdGroupOperation objects

4) Send one mutate request with three operation types

```
$googleAdsServiceClient = $googleAdsClient->getGoogleAdsServiceClient();  
$response = $googleAdsServiceClient->mutate($customerId, $mutateOperations);
```

Creating a new campaign and multiple ad groups of the campaign in one request

PHP

```
private static function buildCampaignBudgetOperation(int $customerId)
{
    return new CampaignBudgetOperation([
        'create' => new CampaignBudget([
            'resource_name' => ResourceNames::forCampaignBudget($customerId, -1),
            'name' => 'Interplanetary Cruise #' . uniqid(),
            'amount_micros' => 500000000
        ])
    ]);
}
```

- Use ResourceNames to help you create a resource name for any objects
- Temporary ID (-1) is hard-coded for simplicity

Creating a new campaign and multiple ad groups of the campaign in one request

```
private static function buildCampaignOperation(  
    int $customerId,  
    string $campaignBudgetResourceName  
) {  
    return new CampaignOperation([  
        'create' => new Campaign([  
            'resource_name' => ResourceNames::forCampaign($customerId, -2),  
            'name' => 'Interplanetary Cruise #' . uniqid(),  
            'advertising_channel_type' => AdvertisingChannelType::SEARCH,  
            'manual_cpc' => new ManualCpc(),  
            'status' => CampaignStatus::PAUSED,  
            'campaign_budget' => $campaignBudgetResourceName,  
        ])  
    ]);  
}
```

- Use ResourceNames to help you create a resource name for any objects
- Temporary ID (-2) is hard-coded for simplicity
- Temporary resource name of the campaign budget is specified here

PHP

Creating a new campaign and multiple ad groups of the campaign in one request

PHP

```
private static function buildAdGroupOperations(string $campaignResourceName)
{
    $operations = [];
    for ($i = 0; $i < 2; $i++) {
        $adGroup = new AdGroup([
            'name' => 'Earth to Mars Cruises #' . uniqid(),
            'campaign' => $campaignResourceName,
            'cpc_bid_micros' => 10000000
        ]);
        $operations[] = new AdGroupOperation(['create' => $adGroup]);
    }
    return $operations;
}
```

Creating a new campaign and multiple ad groups of the campaign in one request

Proprietary + Confidential

PHP

```
$mutateOperations = [];  
$campaignBudgetOperation = self::buildCampaignBudgetOperation($customerId);  
$mutateOperations[] = new MutateOperation(['campaign_budget_operation' => $campaignBudgetOperation]);  
  
$campaignOperation = self::buildCampaignOperation(  
    $customerId,  
    $campaignBudgetOperation->getCreate()->getResourceName()  
);  
$mutateOperations[] = new MutateOperation(['campaign_operation' => $campaignOperation]);  
  
$adGroupOperations = self::buildAdGroupOperations($campaignOperation->getCreate()->getResourceName());  
$mutateOperations = array_merge($mutateOperations, array_map(  
    function (AdGroupOperation $adGroupOperation) {  
        return new MutateOperation(['ad_group_operation' => $adGroupOperation]);  
    },  
    $adGroupOperations  
));
```

```
$googleAdsServiceClient = $googleAdsClient->getGoogleAdsServiceClient();  
$response = $googleAdsServiceClient->mutate($customerId, $mutateOperations);
```

Send a request to create all objects in \$mutateOperations

Batch processing code example

Pausing 500,000 ad group ads

(Based on [AddCompleteCampaignsUsingBatchJob](#))

PHP

```
$batchJobServiceClient = $googleAdsClient->getBatchJobServiceClient();
```

- ① `$batchJobResourceName = self::createBatchJob($batchJobServiceClient, $customerId);`
`self::addAllBatchJobOperations(`
 - `$batchJobServiceClient,`
 - ② `$customerId,`
 - `$batchJobResourceName``);`
- ③ `$operationResponse = self::runBatchJob($batchJobServiceClient, $batchJobResourceName);`
- ④ `self::pollBatchJob($operationResponse);`
- ⑤ `self::fetchAndPrintResults($batchJobServiceClient, $batchJobResourceName);`

Pausing 500,000 ad group ads

(Based on [AddCompleteCampaignsUsingBatchJob](#))

PHP

```
$batchJobServiceClient = $googleAdsClient->getBatchJobServiceClient();
```

- ①

```
$batchJobResourceName = self::createBatchJob($batchJobServiceClient, $customerId);  
self::addAllBatchJobOperations(  
    $batchJobServiceClient,  
    $customerId,  
    $batchJobResourceName  
);
```
- ②
- ③

```
$operationResponse = self::runBatchJob($batchJobServiceClient, $batchJobResourceName);
```
- ④

```
self::pollBatchJob($operationResponse);
```
- ⑤

```
self::fetchAndPrintResults($batchJobServiceClient, $batchJobResourceName);
```


Pausing 500,000 ad group ads

(Based on [AddCompleteCampaignsUsingBatchJob](#))

PHP

```
private static function addAllBatchJobOperations(
    BatchJobServiceClient $batchJobServiceClient,
    int $customerId,
    string $batchJobResourceName
) {
    $response = $batchJobServiceClient->addBatchJobOperations(
        $batchJobResourceName,
        // self::buildAllOperations($customerId)
        self::buildAdGroupAdMutateOperations($customerId, self::$adGroupId, self::$adIds)
    );
    printf(
        "%d mutate operations have been added so far.%s",
        $response->getTotalOperations(),
        PHP_EOL
    );
    // You can use this next sequence token for calling addBatchJobOperations() next time.
    printf(
        "Next sequence token for adding next operations is '%s'.%s",
        $response->getNextSequenceToken(),
        PHP_EOL
    );
}
```

Pausing 500,000 ad group ads

(Based on [AddCompleteCampaignsUsingBatchJob](#))

PHP

```
private static function buildAdGroupAdMutateOperations(  
    string $customerId,  
    string $adGroupId,  
    array $adIds  
) {  
    $mutateOperations = [];  
    foreach ($adIds as $adId) {  
        // Creates an ad group ad.  
        $adGroupAd = new AdGroupAd([  
            'resource_name' => ResourceNames::forAdGroupAd($customerId, $adGroupId, $adId),  
            'status' => AdGroupAdStatus::PAUSED  
        ]);  
  
        // Creates an ad group ad operation and add it to the operations list.  
        $adGroupAdOperation = new AdGroupAdOperation([  
            'update' => $adGroupAd,  
            'update_mask' => FieldMasks::allSetFieldsOf($adGroupAd)  
        ])  
        $mutateOperations[] = new MutateOperation(['ad_group_ad_operation' => $adGroupAdOperation]);  
    }  
    return $mutateOperations;  
}
```

Use the `FieldMasks` utility to help us create a field mask

Caveats & special cases

Caveats & special cases

- Concurrent mutates
 - Mutates *cannot* be executed *concurrently* on the **same object** by more than one source
 - Many applications, such as, Google Ads API, Google Ads UI, modify the same object
 - Many users modify the same object using the Google Ads API
 - The same application modifies the same object using multiple threads
 - If two sources try to simultaneously mutate an object, the API raises a [DatabaseError.CONCURRENT_MODIFICATION_ERROR](#)
- Synchronous mutates may **not always be complete** after the connection ends
 - Uploading conversions (some validations and processing are needed)
 - Adding [Customer Match](#) via [OfflineUserDataJobService](#)
- Some features may have their own limitations & requirements, such as, [remarketing](#), [Hotel ads](#)

Useful References

Resources

Documentation, Guides, Links

[Google Ads API Developer Site](https://developers.google.com/google-ads/api/docs/start)

<https://developers.google.com/google-ads/api/docs/start>

API Structure

<https://developers.google.com/google-ads/api/docs/concepts/api-structure>

Changing and Inspecting Objects

<https://developers.google.com/google-ads/api/docs/concepts/changing-objects>

Mutating Resources

<https://developers.google.com/google-ads/api/docs/mutating/overview>

Batch Processing

<https://developers.google.com/google-ads/api/docs/batch-processing/overview>

Client Libraries & Code Examples

<https://developers.google.com/google-ads/api/docs/client-libs>

Getting Support

Google Ads Developer Blog

<https://ads-developers.googleblog.com>

Google Ads API Forum

<https://groups.google.com/forum/#!forum/adwords-api>

Dedicated Support

googleadsapi-support@google.com